L Number	Hits	Search Text	DB	Time stamp
9	171	pareto	USPAT	2004/04/20 15:59
10	147	pareto and @ad<=20010126	USPAT	2004/04/20 16:00
11	9	(pareto and @ad<=20010126) and (linear adj programming)	USPAT	2004/04/20 16:05
12	3	((pareto and @ad<=20010126) and (linear adj programming)) and cell	USPAT	2004/04/20 16:05
-	580	(print$4 with (workflow jobs)) and cells and @ad<=20010126	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 05:51
-	32	((print$4 with (workflow jobs)) and cells and @ad<=20010126) and (cell with job)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/18 13:38
-	32	(((print$4 with (workflow jobs)) and cells and @ad<=20010126) and (cell with job)) and cell	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/18 13:41
-	9	((((print$4 with (workflow jobs)) and cells and @ad<=20010126) and (cell with job)) and cell) and queu$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/18 13:42
-	3	(((((print$4 with (workflow jobs)) and cells and @ad<=20010126) and (cell with job)) and cell) and queu$4) and (search$4 with (job cell))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/18 14:01
-	2	((((((print$4 with (workflow jobs)) and cells and @ad<=20010126) and (cell with job)) and cell) and queu$4) and (search$4 with (job cell))) and document	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/18 14:02
-	1	(((((((print$4 with (workflow jobs)) and cells and @ad<=20010126) and (cell with job)) and cell) and queu$4) and (search$4 with (job cell))) and document) and bids	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/18 14:02
-	1	((((((print$4 with (workflow jobs)) and cells and @ad<=20010126) and (cell with job)) and cell) and queu$4) and (search$4 with (job cell))) and pending	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/18 14:02
-	2	((((((print$4 with (workflow jobs)) and cells and @ad<=20010126) and (cell with job)) and cell) and queu$4) and (search$4 with (job cell))) and parameters	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/18 14:02
-	3	((((((print$4 with (workflow jobs)) and cells and @ad<=20010126) and (cell with job)) and cell) and queu$4) and (search$4 with (job cell))) and tim$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/18 14:03
-	2	((((((((print$4 with (workflow jobs)) and cells and @ad<=20010126) and (cell with job)) and cell) and queu$4) and (search$4 with (job cell))) and parameters) and tim$4) and module	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/18 14:04
-	1	(((((((((print$4 with (workflow jobs)) and cells and @ad<=20010126) and (cell with job)) and cell) and queu$4) and (search$4 with (job cell))) and parameters) and tim$4) and module) and matrix	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/18 14:04

-	2	(((((((((print$4 with (workflow jobs)) and cells and @ad<=20010126) and (cell with job)) and cell) and queu$4) and (search$4 with (job cell))) and parameters) and tim$4) and (ID identif$8)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/18 14:04
-	2	((((((((print$4 with (workflow jobs)) and cells and @ad<=20010126) and (cell with job)) and cell) and queu$4) and (search$4 with (job cell))) and parameters) and tim$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/18 14:04
-	23	((print$4 with (workflow jobs)) with cells) and @ad<=20010126	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 05:51
-	427	(print$4 and cell and matrix).ab.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 08:59
-	131	((print$4 and cell and matrix).ab.) and ((number ID identif$8) with (cell matrix))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 09:42
-	117	(((print$4 and cell and matrix).ab.) and ((number ID identif$8) with (cell matrix))) and @ad<=20010126	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 10:51
-	1	((((print$4 and cell and matrix).ab.) and ((number ID identif$8) with (cell matrix))) and @ad<=20010126) and binary and decimal and hex	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 09:05
-	1	(((((print$4 and cell and matrix).ab.) and ((number ID identif$8) with (cell matrix))) and @ad<=20010126) and binary and decimal and hex) and (print$4 with (task job process thread))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 09:05
-	11	((print$4 and cell and matrix).ab.) and ((ID identification identifier) with (cell matrix))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 09:18
-	17	((ID identification identifier) with (cell matrix job) with print$4) and print$4 and (cell with matrix)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 09:19
-	10	(((ID identification identifier) with (cell matrix job) with print$4) and print$4 and (cell with matrix)) and @ad<=20010126	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 09:19
-	31	((((print$4 and cell and matrix).ab.) and ((number ID identif$8) with (cell matrix))) and @ad<=20010126) and ((add$4 insert$4 append$4 prepend$4) with (matrix cell))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 09:35
-	43	((print$4 and cell and matrix).ab.) and ((number ID identif$8) with (cell matrix)) and (assign$4 with number ID identif$8)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 10:05
-	1	("6449491").PN.	USPAT	2004/04/19 10:43
-	17941	(assign$4 with (ID identifier identification cell matrix)) and print$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 10:50

-	11236	((assign$4 with (ID identifier identification cell matrix)) and print$4) and @ad<=20010126	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 10:51
-	603	(((assign$4 with (ID identifier identification cell matrix)) and print$4) and @ad<=20010126) and (matrix with cell)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 10:52
-	13967	"603" and 7$/$.ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 10:57
-	162	((((assign$4 with (ID identifier identification cell matrix)) and print$4) and @ad<=20010126) and (matrix with cell)) and 7$/$.ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 10:59
-	9	(((((assign$4 with (ID identifier identification cell matrix)) and print$4) and @ad<=20010126) and (matrix with cell)) and 7$/$.ccls.) and print$4.ab.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 10:59
-	317	(schedul$4 and job and print$4).ab.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 12:01
-	1	((schedul$4 and job and print$4).ab.) and (linear adj program$4)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 12:01
-	23	(linear adj program$4) and 718/1-108.ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 12:02
-	5	((linear adj program$4) and 718/1-108.ccls.) and print$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 14:29
-	553	(linear adj program$4) and print$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 14:30
-	220	((linear adj program$4) and print$4) and schedul$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 14:31
-	170	(((linear adj program$4) and print$4) and schedul$4) and 7$/$.ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 14:32
-	8	((((linear adj program$4) and print$4) and schedul$4) and 7$/$.ccls.) and print$4.ab.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 14:42
-	40	(schedul$4 with (print$4 devices)) and (linear adj programming)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/04/19 14:42

-	16	((schedul$4 with (print$4 devices)) and (linear adj programming)) and @ad<=20010126	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM TDB	2004/04/19 14:43

Groups **Search result 1** for **pareto job optimization OR optimizing OR optimizes "linear programming"**

From: John Gregory (jwg@cray.com) Search Result 1
Subject: **Linear Programming** FAQ
Newsgroups: news.answers, sci.answers, sci.op-research View: Complete Thread (52 articles)
Date: 1993-12-09 10:45:02 PST Original Format

```
Posted-By: auto-faq 2.4
Archive-name: linear-programming-faq
Last-modified: December 1, 1993

         Linear Programming - Frequently Asked Questions List
                    (linear-programming-faq)
          Posted monthly to Usenet newsgroup sci.op-research
                Most recent update: December 1, 1993

----------------------------------------------------------------------

"The best way to get information on Usenet isn't to ask a question, but
 to post the wrong information."  -- aahz@netcom.com

Q0.  "What's in this FAQ?"

A:  Table of Contents
   Q1.  "What is Linear Programming?"
   Q2.  "Where is there a good code to solve LP problems?"
   Q3.  "Oh, and we also want to solve it as an integer program."
   Q4.  "I wrote an optimization code.  Where are some test models?"
   Q5.  "What is MPS format?"
   Q6.  "Just a quick question..."
   Q7.  "What references are there in this field?"
   Q8.  "How do I access the Netlib server?"
   Q9.  "Who maintains this FAQ list?"

See also the related FAQ on Nonlinear Programming (NLP).

----------------------------------------------------------------------

Q1.  "What is Linear Programming?"

A:  A Linear Program (LP) is a problem that can be put into the form

    minimize   cx
    subject to Ax  = b
                x >= 0
```

where x is the vector of variables to be solved for, A is a matrix of known coefficients, and c and b are vectors of known coefficients. The expression "cx" is called the objective function, and the equations "Ax=b" are called the constraints. All these entities must have consistent dimensions, of course, and you can add "transpose" symbols to taste. The matrix A is generally not square, hence you don't solve an LP by just inverting A. Usually A has more columns than rows, and Ax=b is therefore under-determined, leaving great latitude in the choice of x with which to minimize cx.

LP problems are usually solved by a technique known as the Simplex Method, developed in the 1940's and after. Briefly stated, this method works by taking a sequence of square submatrices of A and solving for x, in such a way that successive solutions always improve, until a point is reached where improvement is no longer possible. A family of LP algorithms known as Interior Point (or Barrier) methods comes from nonlinear programming approaches proposed in 1958 and further developed in the late 80's. These methods can be faster for many (but so far not all) large-scale problems. Such methods are characterized by constructing a sequence of trial solutions that go through the interior of the solution space, in contrast to the Simplex Method which stays on the boundary and examines only the corners (vertices). Large-scale LP codes, whatever the algorithm, invariably use sparse matrix techniques.

LP has a variety of uses, in such areas as petroleum, finance, forestry, transportation, and the military.

The word "Programming" is used here in the sense of "planning"; the necessary relationship to computer programming was incidental to the choice of name. Hence the phrase "LP program" to refer to a piece of software is not a redundancy, although I tend to use the term "code" instead of "program" to avoid the possible ambiguity.

Q2. "Where is there a good code to solve LP problems?"

A: Nowadays, with good commercial software (i.e., code that you pay for), models with a few thousand constraints and several thousand variables can be tackled on a 386 PC. Workstations can often handle models with variables in the tens of thousands, or even greater, and mainframes can go larger still. Public domain (free) codes can be relied on to solve models of somewhat smaller dimension. The choice of code can make more difference than the choice of computer hardware. It's hard to be specific about model sizes and speed, a priori, due to the wide variation in things like model structure and variation in factorizing the basis matrices; just because a given code has solved a model of a certain dimension, it may not be able to solve *all* models of the same size, or in the same amount of time.

There is a public domain code, written in C, called "lp_solve" that its author (Michel Berkelaar, email at michel@es.ele.tue.nl) says has solved models with up to 30,000 variables and 50,000 constraints. The author requests that people retrieve it by anonymous ftp from "ftp.es.ele.tue.nl" in directory pub/lp_solve. There is an older version to be found in the Usenet archives, but it contains bugs that have been fixed in the meantime, and hence is unsupported. (As an editorial opinion, I must state that difficult models will give this code trouble. It's not as good as a commercial code. But for someone

who isn't sure just what kind of LP code is needed, it represents a very reasonable first try, since it does solve non-trivial-sized models, and it is free.) The author also made available a program that converts data files from MPS-format into lp_solve's own input format; it's in the same directory, in file mps2eq_0.1.tar.Z.

For academic users only, on a limited variety of platforms, there is available a free version of LOQO, a linear/quadratic program solver. Binary executables have been installed in /pub/opt-net/software/loqo, via anonymous ftp at elib.zib-berlin.de. There are versions for workstations by IBM, Silicon Graphics, HP, and Sun. The package includes a subroutine library (libloqo.a), an executable (loqo), the source for the main part of loqo (loqo.c), and associated documentation (loqo.dvi and *.eps). The algorithm used is a one-phase primal-dual-symmetric interior-point method. If you wish to purchase a commercial version, please contact Bob Vanderbei (rvdb@Princeton.EDU) for details.

The next several suggestions are for public-domain codes that are severely limited by the algorithm they use (tableau Simplex); they may be OK for models with (on the order of) 100 variables and constraints, but it's unlikely they will be satisfactory for larger models. 1) For DOS/PC users, there is an LP and Linear Goal Programming binary called "tslin", by anonymous ftp at garbo.uwasa.fi in directory /pc/ts (the current file name is tslin33b.zip, using ZIP compression), or else I suggest contacting Prof. Salmi at ts@uwasa.fi . For North American users, the garbo server is mirrored on ftp site wuarchive.wustl.edu, in directory mirrors/garbo.uwasa.fi . 2) Also on the garbo server is a file called lp261.zip, having a descriptor of "**Linear Programming** Optimizer by ScanSoft". It consists of PC binaries, and is evidently some sort of shareware (i.e., not strictly public domain). 3) There is an ACM TOMS routine for LP, #552, available from the netlib server, in directory /netlib/toms. This routine was designed for fitting data to linear constraints using an L1 norm, but it uses a modification of the Simplex Method and could presumably be modified to satisfy LP purposes. 4) There are books that contain source code for the Simplex Method. See the section on references. You should not expect such code to be robust. In particular, you can check whether it uses a 2-dimensional array for the A-matrix; if so, it is surely using the tableau Simplex Method rather than sparse methods, and it will not be useful for large models.

The following suggestions may represent a low-cost way of solving LP's if you already have certain software available to you. 1) Some spreadsheet programs have an embedded LP solver, or offer one as an installable option. 2) A package called QSB (Quantitative Systems for Business, from Prentice-Hall publishers) has an LP module among its routines. 3) If you have access to a commercial math library, such as IMSL or NAG, you may be able to use an LP routine from there.
4) Mathematical systems MATLAB (The Math Works, Inc., (508) 653-1415) and MAPLE (reference?) also have LP solvers; an interface from MATLAB to lp_solve is available from Jeff Kantor (Jeffrey.Kantor@nd.edu), and there's also a Simplex code written in the MATLAB language, available from the netlib server, file netlib/matlab/**optimization**/simplex1.m.Z. (There's a Usenet newsgroup on MATLAB: comp.soft-sys.matlab.) If speed matters to you, then according to a Usenet posting by Pascal Koiran (koiran@ens-lyon.fr), on randomly generated LP models, MATLAB was an order of magnitude faster than MAPLE on a 200x20 problem but an order of magnitude slower than lp_solve on a 50x100 problem. (I don't intend to get into benchmarking in this document, but I mention these results just to explain why I choose to focus mostly on special purpose

LP software.)

If your models prove to be too difficult for free or add-on software to handle, then you may have to consider acquiring a commercial LP code. Dozens of such codes are on the market. There are many considerations in selecting an LP code. Speed is important, but LP is complex enough that different codes go faster on different models; you won't find a "Consumer Reports" article to say with certainty which code is THE fastest. I usually suggest getting benchmark results for your particular type of model if speed is paramount to you. Benchmarking can also help determine whether a given code has sufficient numerical stability for your kind of models.

Other questions you should answer: Can you use a stand-alone code, or do you need a code that can be used as a callable library, or do you require source code? Do you want the flexibility of a code that runs on many platforms and/or operating systems, or do you want code that's tuned to your particular hardware architecture (in which case your hardware vendor may have suggestions)? Is the choice of algorithm (Simplex, Interior Point) important to you? Do you need an interface to a spreadsheet code? Is the purchase price an overriding concern? If you are at a university, is the software offered at an academic discount? How much hotline support do you think you'll need? There is usually a large difference in LP codes, in performance (speed, numerical stability, adaptability to computer architectures) and in features, as you climb the price scale.

At the end of this section is a *very* condensed version of a survey of LP software published in the June 1992 issue of "OR/MS Today", a joint publication of ORSA (Operations Research Society of America) and TIMS (The Institute of Management Science). For further information I would suggest you read the full article. It's likely that you can find a copy, either through a library, or by contacting a member of these two organizations (most universities probably have several members among the faculty and student body). This magazine also carries advertisements for many of these products, which may give you additional information to help make a decision.

The author of that survey, Ramesh Sharda, has updated and expanded it for 1993 into a larger report called "Linear and Discrete **Optimization** and Modeling Software: A Resource Handbook". For information, contact Lionheart Publishing Inc., 2555 Cumberland Parkway, Suite 299, Atlanta, GA 30339. Phone: (404)-431-0867. This book is fairly expensive, and geared toward users whose needs for LP software are considerable. Another book, to be available in November 1993 is "**Optimization** Software Guide," by Jorge More and Stephen Wright, from SIAM Books. Call 1-800-447-7426 to order ($24.50, less ten percent if you are a SIAM member). It sounds promising ("75 software packages...").

In the table below, I give the name of the software and the phone number listed in the June 1992 survey. I have included, where I know of one, an email address (information not given in the June 1992 survey), and other information obtained from non-proprietary sources. To keep the table short enough to fit here, I decided not to include postal addresses. (I suppose that an "800" number will not be useful to people outside the US; consult the full survey for more information on contacting such vendors. Also, for some companies there may exist European or Asian contact points, but this is beyond the scope of this document.)

The first part of the table consists of software I deem to be LP solvers. The second part is software that in some sense is a front end to the solvers (modeling software). In some cases it becomes a hazy distinction, but I hope this arrangement of information turns out to be useful to the reader.

Under "H/W" is the minimum hardware said to be needed to run the code; generally I conceive of a hierarchy where PC's (and Macintoshes) are the least powerful, then workstations (WS) like Suns and RS-6000's, on up to supercomputers, so by the symbol "^" I mean "and up", namely that most commonly-encountered platforms are supported beyond the given minimum level.

Even more so than usual, I emphasize that you must use this information at your own risk. I provide it as a convenience to those readers who have difficulty in locating the OR/MS Today survey, I take no responsibility for errors either in the original article or by my act of copying it manually, though I will gladly correct any mistakes that are pointed out to me.

```
Key to Features:  S=Simplex     I=Interior-Point or Barrier
                  Q=Quadratic   G=General-Nonlinear
                  M=MIP         N=Network
                  V=Visualization
```

Solver Code Name	Feat.	H/W	Phone	Email address
AT&T KORBX	IQ	WS ^	908-949-8966	
Best Answer	S	Mac-Plus	510-943-7667	
CPLEX	SIMN	PC-386 ^	702-831-7744	info@cplex.com
Excel	SMG	PC/Mac	206-882-8080	
FortLP	SM	PC ^	708-971-2337	
HS/LP	SM	PC-386/VAX	201-627-1424	
INCEPTA	SMV	PC-386	416-896-0515	
LAMPS	SM	PC-386 ^	413-584-1605	al@andltd.and.nl
LINDO	SMQ	PC ^	800-441-2378	lindo@delphi.com
LOQO	IQ	PC ^	609-258-0876	rvdb@princeton.edu
LP88	S	PC	703-360-7600	
MathPro	SMV	PC-286/WS	202-887-0296	
MILP88	SM	PC	703-360-7600	
MILP LO	SV	PC	(+361)149-7531	
MPS-III	SMNQ	PC-386 ^	703-558-8701	
MSLP-PC	S	PC	604-361-9778	
OMP	SM	PC/VAX/WS	919-847-9997	
OSL	SIMNQ	PC/WS/IBM	914-385-6034	randym@vnet.ibm.com
PC-PROG	SMQ	PC	919-847-9997	Ge.vanGeldorp@lr.tudelft.nl
SAS/OR	SMN	PC ^	919-677-8000	
SCICONIC	SM	PC-386 ^	(+44)908-585858	
STORM	SMN	PC	216-464-1209	
TurboSimplex	S	PC/Mac	703-351-5272	
What If	SMG	PC	800-447-2226	
XA	SM	PC ^	818-441-1565	sunsetsoft@aol.com
XPRESS-MP	SM	PC-286 ^	202-887-0296	
YLP	S	PC ^	702-831-4967	

Modeling Code Name	H/W	Phone	Email address
DATAFORM	PC-386 ^	703-558-8701	
GAMS	PC-286 ^	415-583-8840	gams@gams.com

```
LINGO            PC ^        800-441-2378 lindo@delphi.com
MIMI/LP          WS          908-464-8300
MPL Sys.         PC          703-351-5272
OMNI             PC-386 ^    202-627-1424
VMP              PC-386/WS   301-622-0694
What's Best!     PC/Mac/WS   800-441-2378 lindo@delphi.com
```

Q3. "Oh, and we also want to solve it as an integer program.

A: Integer LP models are ones where the answers must not take fractional values. It may not be obvious that this is a VERY much harder problem than ordinary LP, but it is nonetheless true. The buzzword is "NP-Completeness", the definition of which is beyond the scope of this document but means in essence that, in the worst case, the amount of time to solve a family of related problems goes up exponentially as the size of the problem grows, for all algorithms that solve such problems to a proven answer.

Integer models may be ones where only some of the variables are to be integer and others may be real-valued (termed "Mixed Integer LP" or MILP, or "Mixed Integer Programming" or MIP); or they may be ones where all the variables must be integral (termed "Integer LP" or ILP). The class of ILP is often further subdivided into problems where the only legal values are {0,1} ("Binary" or "Zero-One" ILP), and general integer problems. For the sake of generality, the Integer LP problem will be referred to here as MIP, since the other classes can be viewed as special cases of MIP. MIP, in turn, is a particular member of the class of Discrete **Optimization** problems.

People are sometimes surprised to learn that MIP problems are solved using floating point arithmetic. Although various algorithms for MIP have been studied, most if not all available general purpose large-scale MIP codes use a method called "Branch and Bound" to try to find an optimal solution. Briefly stated, B&B solves MIP by solving a sequence of related LP models. (As a point of interest, the Simplex Method currently retains an advantage over the newer Interior Point methods for solving these sequences of LP's.) Good codes for MIP distinguish themselves more by solving shorter sequences of LP's, than by solving the individual LP's faster. Even more so than with regular LP, a costly commercial code may prove its value to you if your MIP model is difficult.

You should be prepared to solve *far* smaller MIP models than the corresponding LP model, given a certain amount of time you wish to allow (unless you and your model happen to be very lucky). There exist models that are considered challenging, with a few dozen variables. Conversely, some models with tens of thousands of variables solve readily. The best explanations of "why" usually seem to happen after the fact. 8v) But a MIP model with hundreds of variables should always be approached, initially at least, with a certain amount of humility.

A major exception to this somewhat gloomy outlook is that there are certain models whose LP solution always turns out to be integer. Best known of these is the so-called Network-Flow Problem. Special cases of this problem are the Transportation Problem, the Assignment Problem, and the Shortest Path Problem. The theory of unimodular matrices is fundamental here. It turns out that these particular problems are best

solved by specialized routines that take major shortcuts in the Simplex Method, and as a result are relatively quick-running even compared to ordinary LP. Some commercial LP solvers include a network solver. See [Kennington], which contains some source code for Netflo. Netflo is available by anonymous ftp at dimacs.rutgers.edu, in directory

    /pub/netflow/mincost/solver-1

but I don't know the copyright situation (I always thought you had to buy the book to get the code). Another text containing Fortran code is [Bertsekas], though I am unaware of any place that has the source code online. There is an ACM TOMS routine, #548, that solves the Assignment problem using the Hungarian Method, available from the netlib server, in directory /netlib/toms. An article in the ORSA Journal on Computing in 1991 by Kennington and Wang investigated the performance of some algorithms.

The public domain code "lp_solve", mentioned earlier, accepts MIP models, as do a large number of the commercial LP codes in the OR/MS Today survey (see section above). I have seen mention made of algorithm 333 in the Collected Algorithms from CACM, though I'd be surprised if it was robust enough to solve large models.

In [Syslo] is code for 28 algorithms, most of which pertain to some aspect of Discrete **Optimization**.

There is a code called Omega that analyzes systems of linear equations in integer variables. It does not solve **optimization** problems, except in the case that a model reduces completely, but its features could be useful in analyzing and reducing MIP models. Have a look via anonymous ftp at ftp.cs.umd.edu:pub/omega (documentation is provided there), or contact Bill Pugh at pugh@cs.umd.edu.

Mustafa Akgul (AKGUL@TRBILUN.BITNET) at Bilkent University maintains an archive via anonymous ftp (firat.bcc.bilkent.edu.tr or 139.179.10.13). In addition to a copy of lp_solve (though I would recommend using the official source listed in the previous section), there is mip386.zip, which is a zip-compressed code for PC's. He also has a couple of network codes and various other codes he has picked up. All this is in directory pub/IEOR/Opt and its further subdirectories LP, PC, and Network.

The better commercial MIP codes have numerous parameters and options to give the user control over the solution strategy. Most have the capability of stopping before an optimum is proved, printing the best answer obtained so far. For many MIP models, stopping early is a practical necessity. Fortunately, a solution that has been proved by the algorithm to be within, say, 1% of optimality often turns out to be the true optimum, and the bulk of the computation time is spent proving the optimality. For many modeling situations, a near-optimal solution is acceptable anyway.

Once one accepts that large MIP models are not typically solved to a proved optimal solution, that opens up a broad area of approximate methods, probabilistic methods and heuristics, as well as modifications to B&B. See [Balas] which contains a useful heuristic for 0-1 MIP models. See also the brief discussion of Genetic Algorithms and Simulated Annealing in the FAQ on Nonlinear Programming.

Whatever the solution method you choose, when trying to solve a difficult MIP model, it is usually crucial to understand the workings of the physical system (or whatever) you are modeling, and try to find

some insight that will assist your chosen algorithm to work better. A related observation is that the way you formulate your model can be as important as the actual choice of solver. You should consider getting some assistance if this is your first time trying to solve a large (>100 integer variable) problem.

Q4. "I wrote an **optimization** code. Where are some test models?"

A: If you want to try out your code on some real-world LP models, there is a very nice collection of small-to-medium-size ones (with a few that are rather large) on netlib, in directory lp/data. The netlib LP files (after you uncompress them) are in a format called MPS, which is described in another section of this document.

Also on netlib is a collection of infeasible LP models, located in directory lp/infeas.

There is a collection of MIP models, housed at Rice University. Send an email message containing "send catalog" to softlib@rice.edu , to get started. Or try anonymous ftp at softlib.cs.rice.edu, then "cd /pub/miplib".

There is a collection of network-flow codes and models at DIMACS (Rutgers University). Use anonymous FTP at dimacs.rutgers.edu. Start looking in /pub/netflow. Another network generator is called NETGEN and is available on netlib (lp/generators).

The modeling language GAMS comes with about 150 test models, which you might be able to test your code with.

John Beasley has posted information on his OR-Lib, which contains various **optimization** test problems. Send e-mail to umtsk99@vaxa.cc.imperial.ac.uk to get started. Or have a look in the Journal of the Operational Research Society, Volume 41, Number 11, Pages 1069-72. Information about test problems for the problem areas listed below can be obtained by emailing o.rlibrary@ic.ac.uk with the email message being the file name for the problem areas you are interested in.

Problem area	File name
Assignment problem	assigninfo
Crew scheduling	cspinfo
Data envelopment analysis	deainfo
Generalised assignment problem	gapinfo
Integer programming	mipinfo
Linear programming	lpinfo
Location:	
capacitated warehouse location	capinfo
p-median	pmedinfo
uncapacitated warehouse location	uncapinfo
Multiple knapsack problem	mknapinfo
Quadratic assignment problem	qapinfo
Resource constrained shortest path	rcspinfo
Scheduling:	
flow shop	flowshopinfo
job shop	jobshopinfo
open shop	openshopinfo

```
  Set covering                                  scpinfo
  Set partitioning                              sppinfo
  Steiner:
       Euclidean Steiner problem                esteininfo
       Rectilinear Steiner problem              rsteininfo
       Steiner problem in graphs                steininfo
  Travelling salesman problem                   tspinfo
  Two-dimensional cutting:
       assortment problem                       assortinfo
       constrained guillotine                   cgcutinfo
       constrained non-guillotine               ngcutinfo
       unconstrained guillotine                 gcutinfo
  Vehicle routing:
       fixed areas                              areainfo
       fixed routes                             fixedinfo
       period routing                           periodinfo
       single period                            vrpinfo
```

Q5. "What is MPS format?"

A: MPS format was named after an early IBM LP product and has emerged as a de facto standard ASCII medium among most of the commercial LP codes. Essentially all commercial LP codes accept this format, but if you are using public domain software and have MPS files, you may need to write your own reader routine for this. It's not too hard. See also the comment regarding the lp_solve code, in another section of this document, for the availability of an MPS reader.

The main things to know about MPS format are that it is column oriented (as opposed to entering the model as equations), and everything (variables, rows, etc.) gets a name. MPS format is described in more detail in [Murtagh].

MPS is an old format, so it is set up as though you were using punch cards, and is not free format. Fields start in column 1, 5, 15, 25, 40 and 50. Sections of an MPS file are marked by so-called header cards, which are distinguished by their starting in column 1. Although it is typical to use upper-case throughout the file (like I said, MPS has long historical roots), many MPS-readers will accept mixed-case for anything except the header cards, and some allow mixed-case anywhere. The names that you choose for the individual entities (constraints or variables) are not important to the solver; you should pick names that are meaningful to you, or will be easy for a post-processing code to read.

Here is a little sample model written in MPS format (explained in more detail below):

```
NAME          TESTPROB
ROWS
 N  COST
 L  LIM1
 G  LIM2
 E  MYEQN
COLUMNS
    XONE      COST                 1   LIM1                 1
    XONE      LIM2                 1
    YTWO      COST                 4   LIM1                 1
```

```
    YTWO      MYEQN              -1
    ZTHREE    COST                9   LIM2                1
    ZTHREE    MYEQN               1
RHS
    RHS1      LIM1                5   LIM2               10
    RHS1      MYEQN               7
BOUNDS
 UP BND1      XONE                4
 LO BND1      YTWO               -1
 UP BND1      YTWO                1
ENDATA
```

For comparison, here is the same model written out in an equation-oriented format:

```
Optimize
 COST:    XONE + 4 YTWO + 9 ZTHREE
Subject To
 LIM1:    XONE + YTWO <= 5
 LIM2:    XONE + ZTHREE >= 10
 MYEQN:   - YTWO + ZTHREE  = 7
Bounds
 0 <= XONE <= 4
-1 <= YTWO <= 1
End
```

Strangely, there is nothing in MPS format that specifies the direction of **optimization**. And there really is no standard "default" direction; some LP codes will maximize if you don't specify otherwise, others will minimize, and still others put safety first and have no default and require you to specify it somewhere in a control program or by a calling parameter. If you have a model formulated for minimization and the code you are using insists on maximization (or vice versa), it may be easy to convert: just multiply all the coefficients in your objective function by (-1). The optimal value of the objective function will then be the negative of the true value, but the values of the variables themselves will be correct.

The NAME card can have anything you want, starting in column 15. The ROWS section defines the names of all the constraints; entries in column 2 or 3 are E for equality rows, L for less-than (<=) rows, G for greater-than (>=) rows, and N for non-constraining rows (the first of which would be interpreted as the objective function). The order of the rows named in this section is unimportant.

The largest part of the file is in the COLUMNS section, which is the place where the entries of the A-matrix are put. All entries for a given column must be placed consecutively, although within a column the order of the entries (rows) is irrelevant. Rows not mentioned for a column are implied to have a coefficient of zero.

The RHS section allows one or more right-hand-side vectors to be defined; most people don't bother having more than one. In the above example, the name of the RHS vector is RHS1, and has non-zero values in all 3 of the constraint rows of the problem. Rows not mentioned in an RHS vector would be assumed to have a right-hand-side of zero.

The optional BOUNDS section lets you put lower and upper bounds on individual variables (no * wild cards, unfortunately), instead of having to define extra rows in the matrix. All the bounds that have

a given name in column 5 are taken together as a set. Variables not mentioned in a given BOUNDS set are taken to be non-negative (lower bound zero, no upper bound). A bound of type UP means an upper bound is applied to the variable. A bound of type LO means a lower bound is applied. A bound type of FX ("fixed") means that the variable has upper and lower bounds equal to a single value. A bound type of FR ("free") means the variable has neither lower nor upper bounds.

There is another optional section called RANGES that I won't go into here. The final card must be ENDATA, and yes, it is spelled funny.

Q6. "Just a quick question..."

Q: What is a matrix generator?
A: This is a code that creates input for an LP (or MIP, or NLP) code, using a more natural input than MPS format. There are no free ones. Matrix generators can be roughly broken into two classes, column oriented ones, and equation oriented ones. The former class is older, and includes such commercial products as OMNI (Haverley Systems) and DATAFORM (Ketron). Big names in the latter class are GAMS (Scientific Press), LINGO (LINDO Systems), and AMPL (information is in netlib/opt on the netlib server, or send email to 70742.555@compuserve.com). These products have links to various solvers (commercial and otherwise).

Q: How do I diagnose an infeasible LP model?
A: A model is infeasible if the constraints are inconsistent, i.e., if no feasible solution can be constructed. It's often difficult to track down a cause. The cure may even be ambiguous: is it that some demand was set too high, or a supply set too low? A useful technique is Goal Programming, one variant of which is to include two explicit slack variables (positive and negative), with huge cost coefficients, in each constraint. The revised model is guaranteed to have a solution, and you can look at which rows have slacks that are included in the "optimal" solution. By the way, I recommend a Goal Programming philosophy even if you aren't having trouble with feasibility; "come on, you could probably violate this constraint for a price." 8v) Another approach is Fourier-Motzkin Elimination (article by Danztig and Eaves in the Journal of Combinatorial Theory (A) 14, 288-297 (1973). A software system called ANALYZE was developed by Harvey Greenberg to provide computer-assisted analysis, including rule-based intelligence; for further information about this code, and a bibliography of more than 400 references on the subject of model analysis, contact Greenberg at HGREENBERG@cudnvr.denver.colorado.edu. A system based on the MINOS solver, called MINOS(IIS), available from John Chinneck (chinneck@sce.carleton.ca), can also be used to identify a so-called Irreducible Infeasible Subset. As a final comment, commercial codes sometimes have built-in features to help.

Q: I want to know the specific constraints that contradict each other.
A: This may not be a well posed problem. If by this you mean you want to find the minimal set of constraints that should be removed to restore feasibility, this can be modeled as an Integer LP (which means, it's potentially a harder problem than the underlying LP itself). Start with a Goal Programming approach as outlined above, and introduce some 0-1 variables to turn the slacks off or on. Then minimize on the sum of these 0-1 variables. An article

covering another approach to this question is by Chinneck and Dravnieks in the Spring 1991 ORSA Journal on Computing (vol 3, number 2).

5 Q: I just want to know whether or not a feasible solution *exists*.
A: From the standpoint of computational complexity, finding out if a model has a feasible solution is essentially as hard as finding the optimal LP solution, within a factor of 2 on average, in terms of effort in the Simplex Method. There are no shortcuts in general, unless you know something useful about your model's structure (e.g., if you are solving some form of a transportation problem, you may be able to assure feasibility by checking that the sources add up to at least as great a number as the sum of the destinations).

16 Q: I have an LP, except it's got several objective functions.
A: Fundamental to the class of Multiple Criteria models is that there may no longer be the concept of a unique solution. I am unaware of any public domain code to approach such problems, though I have seen a reference to MATLAB's **Optimization** Toolbox. Approaches that have worked are:

- Goal Programming (treat the objectives as constraints with costed slacks), or, almost equivalently, form a composite function from the given objective functions;
- **Pareto** preference analysis (essentially brute force examination);
- Put your objective functions in priority order, optimize on one objective, then change it to a constraint fixed at the optimal value (perhaps subject to a small tolerance), and repeat with the next function.

There is a section on this whole topic in Reference [1]. As a final piece of advice, if you can cast your model in terms of physical realities, or dollars and cents, sometimes the multiple
33 objectives disappear! 8v)

Q: I have an LP that has large almost-independent matrix blocks that are linked by a few constraints. Can I take advantage of this?
A: In theory, yes. See section 6.2 in Reference [1] for a discussion of Dantzig-Wolfe decomposition. I am told that the commercial code OSL has features to assist in doing this. With any other code, you'll have to create your own framework and then call the LP solver to solve the subproblems. The folklore is that generally such schemes take a long time to converge so that they're slower than just solving the model as a whole, although research continues. For now my advice, unless you are using OSL or your model is so huge that a good solver can't fit it in memory, is to not bother decomposing it. It's probably more cost effective to upgrade your solver, if the algorithm is limiting you, than to invest your time; but I suppose that's an underlying theme in a lot of my advice. 8v)

Q: I need to find all integer points in a polytope.
A: There is no known way of doing this efficiently (i.e., with an algorithm that grows only polynomially with the problem size). For small models, it may be practical to find your answer by complete enumeration. A related question is how to find all the vertices of an LP, with the same pessimistic answer. [Schrijver] is said to discuss this. Two people mentioned on the net, said to be working on code for this, are Dick Helgason (helgason@seas.smu.edu) and Betty Hickman (hickman@odin.unomaha.edu).

```
Q:  Are there any parallel LP codes?
A:  IBM has announced a parallel Branch and Bound capability in their
    package OSL, for use on clusters of workstations.  "This is real,
    live commercial software, not a freebie. Contact
    forrest@watson.ibm.com".  Jeffrey Horn (horn@cs.wisc.edu) recently
    compiled a bibliography of papers relating to research on parallel
    B&B.  There is an annotated bibliography of parallel methods in
    Operations Research in general, in Vol 1 (1), 1989 of the ORSA
    Journal on Computing, although by now it might be a little out of
    date.  I'm not aware of any implementations (beyond the "toy"
    level) of general sparse Simplex or interior-point solvers on
    parallel machines.  If your particular model is a good candidate
    for decomposition (see topic, above) then parallelism could be very
    useful, but you'll have to implement it yourself.

Q:  I am trying to solve a Traveling Salesman Problem ...
A:  TSP is a famously hard problem that has attracted many of the best
    minds in the field.  Look at the bibliography in the Integer
    Programming section of Reference [1], particularly the ones with
    the names Groetschel and/or Padberg in them.  Solving for a proved
    optimum is combinatorial in nature.  There are some heuristics for
    getting a "good" solution; see the article by Lin and Kernighan in
    Operations Research, Vol 21 (1973), pp 498-516.  I don't believe
    there are any commercial products to solve this problem.  [Syslo]
    contains some algorithms and Pascal code.

Q:  I need to do post-optimal analysis.
A:  Many commercial LP codes have features to do this.  Also called
    Ranging or Sensitivity Analysis, it gives information about how the
    coefficients in the problem could change without affecting the
    nature of the solution.  Most LP textbooks, such as Reference [1],
    describe this.  Unfortunately, all this theory applies only to LP.

    For a MIP model with both integer and continuous variables, you
    could get a limited amount of information by fixing the integer
    variables at their optimal values, resolving the model as an LP,
    and doing standard post-optimal analyses on the remaining
    continuous variables; but this tells you nothing about the integer
    variables, which presumably are the ones of interest.  Another MIP
    approach would be to choose the coefficients of your model that are
    of the most interest, and generate "scenarios" using values within
    a stated range created by a random number generator.  Perhaps five
    or ten scenarios would be sufficient; you would solve each of them,
    and by some means compare, contrast, or average the answers that
    are obtained.  Noting patterns in the solutions, for instance, may
    give you an idea of what solutions might be most stable.  A third
    approach would be to consider a goal-programming formulation;
    perhaps your desire to see post-optimal analysis is an indication
    that some important aspect is missing from your model.

Q:  Some versions of the Simplex algorithm require as input a vertex.
    Do all LP codes require a starting vertex?
A:  No.  You just have to give an LP code the constraints and the
    objective function, and it will construct the vertices for you.
    Most codes go through a so-called two phase method, wherein the
    code first looks for a feasible solution, and then works on getting
    an optimal solution.  The first phase can begin anywhere, such as
    with all the variables at zero (though commercial codes typically
    have a so-called "crash" algorithm to pick a better starting
    point).  So, no, you don't have to give a code a starting point.
```

On the other hand, it is not uncommon to do so, because it can speed up the solution time tremendously. Commercial codes usually allow you to do this (they call it a "basis", though that's a loose usage of a specific linear algebra concept); free codes generally don't. You'd normally want to bother with a starting basis only when solving families of related and difficult LP's (i.e., in some sort of production mode).

Q7. "What references are there in this field?"

A: What follows here is an idiosyncratic list, a few books that I like or have been recommended on the net. I have *not* reviewed them all.

Regarding the common question of the choice of textbook for a college LP course, it's difficult to give a blanket answer because of the variety of topics that can be emphasized: brief overview of algorithms, deeper study of algorithms, theorems and proofs, complexity theory, efficient linear algebra, modeling techniques, solution analysis, and so on. An unscientific poll of ORCS-L mailing list readers uncovered a consensus that [Chvatal] was in most ways pretty good, at least for an algorithmically oriented class. For a class in modeling, a book about a commercial code (LINDO, AMPL, GAMS are suggested) would be useful, especially if the students are going to use such a code; and I have always had a fondness for [Williams]. I have marked with an arrow ("->") books that received positive mention in this poll (I included my own votes too 8v)).

General reference [1]
- Nemhauser, Rinnooy Kan, & Todd, eds, **Optimization**, North-Holland, 1989. (Very broad-reaching, with large bibliography. Good reference; it's the place I tend to look first. Expensive, and tough reading for beginners.)

LP textbooks
-> Bazaraa, Jarvis and Sherali. **Linear Programming** and Network Flows. (Grad level.)
-> Chvatal, **Linear Programming**, Freeman, 1983. (Undergrad or grad.)
-> Daellenbach and Bell, A User's Guide to LP. (Good for engineers, but may be out of print.)
-> Ecker & Kupferschmid, Introduction to Operations Research.
- Luenberger, Introduction to Linear and Nonlinear Programming, Addison Wesley, 1984. (Updated version of an old standby.)
-> Murtagh, B., Advanced **Linear Programming**, McGraw-Hill, 1981. (Good one after you've read an introductory text.)
- Murty, K., Linear and Combinatorial Programming.
-> Schrijver, A., Theory of Linear and Integer Programming, Wiley, 1986. (Advanced)
-> Taha, H., Operations Research: An Introduction, 1987.
-> Thie, P.R., An Introduction to **Linear Programming** and Game Theory, Wiley, 1988.
-> Williams, H.P., Model Building in Mathematical Programming, Wiley 1985. (Little on algorithms, but excellent for learning what makes a good model.)

Interior Point LP (popularly but imprecisely called "Karmarkar")
(There is also a bibliography (with over 1300 entries!?!) obtainable by mailing to "netlib@ornl.gov" and saying "send intbib.bib from bib".)
-> Fang and Puthenpura, Linear **Optimization** and Extensions. (Grad

level textbook, also contains some Simplex and Ellipsoid. I heard mixed opinions on this one.)
- Lustig, Marsten & Shanno, "Interior Point Methods for **Linear Programming** - Computational State of the Art", to appear in ORSA Journal on Computing, early 1994. (Available as a tech report from shanno@dantzig.rutgers.edu as RUTCOR Report RRR 41-92.)
- Marsten, et al., "Interior Point Methods for **Linear Programming**", Interfaces, pp 105-116, July-August 1990. (Introductory article, written by authors of a good commercial code, article superseded by [Lustig] when it appears.)

Documentation for commercial codes
-> Brooke, Kendrick & Meeraus, GAMS: A Users' Guide, The Scientific Press, 1988.
-> Fourer, Gay & Kernighan, AMPL: A Modeling Language for Mathematical Programming, The Scientific Press, 1992.
-> Greenberg, H.J., Modeling by Object-Driven Linear Elemental Relations: A User's Guide for MODLER, Kluwer Academic Publishers, 1993.
-> Schrage, L., LINDO: An **Optimization** Modeling System, The Scientific Press, 1991.

Books containing source code

- Arbel, Ami, Exploring Interior-Point **Linear Programming**, MIT Press, 1993. (New, I have no information about the quality of this book. Supposed to include IBM PC software.)
- Best and Ritter, **Linear Programming**: active set analysis and computer programs, Prentice-Hall, 1985.
- Bertsekas, D.P., Linear Network **Optimization**: Algorithms and Codes, MIT Press, 1991.
- Bunday and Garside, **Linear Programming** in Pascal, Edward Arnold Publishers, 1987.
- Bunday, **Linear Programming** in Basic (presumably the same publisher).
- Kennington & Helgason, Algorithms for Network Programming, Wiley, 1980. (A special case of LP; contains Fortran source code.)
- Press, Flannery, Teukolsky & Vetterling , Numerical Recipes, Cambridge, 1986. (Comment: use their LP code with care.)
- Syslo, Deo & Kowalik, Discrete **Optimization** Algorithms with Pascal Programs, Prentice-Hall (1983). (Contains code for 28 algorithms such as Revised Simplex, MIP, networks.)

Other publications
- Ahuja, Magnanti & Orlin, Network Flows, Prentice Hall, 1993.
- Balas, E. and Martin, C., "Pivot And Complement: A Heuristic For 0-1 Programming Problems", Management Science, 1980, Vol 26, pp 86-96.
- Bondy & Murty, Graph Theory with Applications.
- Forsythe, Malcolm & Moler, Computer Methods for Mathematical Computations, Prentice-Hall.
-> Gill, Murray and Wright, Numerical Linear Algebra and **Optimization**, Addison-Wesley, 1991.
- Greenberg, H.J., A Computer-Assisted Analysis System for Mathematical Programming Models and Solutions: A User's Guide for ANALYZE, Kluwer Academic Publishers, 1993.
- Lawler, Lenstra, et al, The Traveling Salesman Problem, Wiley, 1985.
- Murty, Network Programming, Prentice Hall, 1992.
- Reeves, C.R., ed., Modern Heuristic Techniques for Combinatorial Problems, Halsted Press (Wiley). (Contains chapters on tabu search, simulated annealing, genetic algorithms, neural nets, and Lagrangean relaxation.)

Q8. "How do I access the Netlib server?"

A: If you have ftp access, you can try "ftp research.att.com", using "netlib" as the Name, and your email address as the Password. Do a "cd <dir>" where <dir> is whatever directory was mentioned, and look around, then do a "get <filename>" on anything that seems interesting. There often will be a "README" file, which you would want to look at first. Alternatively, you can reach an e-mail server via "netlib@ornl.gov", to which you can send a message saying "send index from <dir>"; follow the instructions you receive.

Q9. "Who maintains this FAQ list?"

A: John W. Gregory jwg@cray.com 612-683-3673
Applications Dept. Cray Research, Inc. Eagan, MN 55121 USA

I suppose I should say something here to the effect that "the material in this document does not reflect any official position taken by Cray Research, Inc." Also, "use at your own risk", "no endorsement of products mentioned", etc., etc. "IMHO"s are implicit throughout.

In compiling this information, I have drawn on my own knowledge of the field, plus information from contributors to Usenet groups and the ORCS-L mailing list. I give my thanks to all those who have offered advice and support. I've tried to keep my own biases (primarily, toward the high end of computing) from dominating what I write here, and other viewpoints that I've missed are welcome. Suggestions, corrections, topics you'd like to see covered, and additional material (particularly on NLP) are solicited.

Copies of this FAQ list may be made freely, as long as it is distributed at no charge and with the date of last update and this disclaimer included. If you wish to cite this FAQ formally (hey, someone actually asked me this), you may use:
Gregory, John W. (1993) "**Linear Programming** FAQ", Usenet sci.answers. Available via anonymous ftp from rtfm.mit.edu in /pub/usenet/sci.answers/**linear-programming**-faq

There's a mail server on that machine, so if you don't have ftp privileges, you can send an e-mail message to mail-server@rtfm.mit.edu containing:
send usenet/sci.answers/**linear-programming**-faq
as the body of the message.

END **linear-programming**-faq